FIG. 1

```
C SOURCE CODE
N=8;
for(i=0; i<N; i++)
      y+=x[i];
```

```
#1.                 LDRS _repeat_start
#2.                 LDRE _repeat_end
#3.                 LDRC #8
#4.                 MOV  #x_addr,r0
#5.                 MOV  #y_addr,r1
#6.                 MOV  #0,r3
#7. _repeat_start   MOV  @r0+,r2
#8. _repeat_end     ADD  r2,r3
#9.                 MOV  r3,@r1
```

FIG. 2

```
LDRS _repeat_start : MAKE REPEAT START PC _repeat_start
LDRE _repeat_end   : MAKE REPEAT END PC _repeat_end
LDRC #N            : MAKE REPEAT NUMBER N
```

FIG. 3

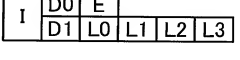


FIG. 4

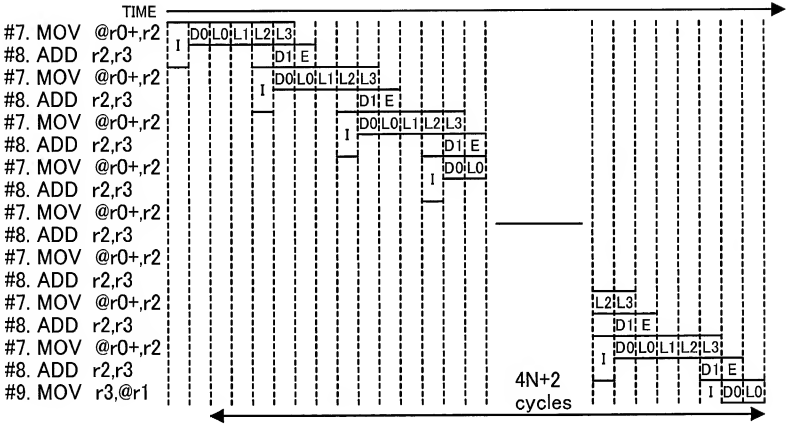

TIME
#7. MOV @r0+,r2
#8. ADD r2,r3
#7. MOV @r0+,r2
#8. ADD r2,r3
#7. MOV @r0+,r2
#8. ADD r2,r3
#7. MOV @r0+,r2
#8. ADD r2,r3
#7. MOV @r0+,r2
#8. ADD r2,r3
#7. MOV @r0+,r2
#8. ADD r2,r3
#7. MOV @r0+,r2
#8. ADD r2,r3
#9. MOV r3,@r1
I
D0 L0 L1 L2 L3
D1 E
D0 L0
L2 L3
4N+2
cycles

FIG. 5

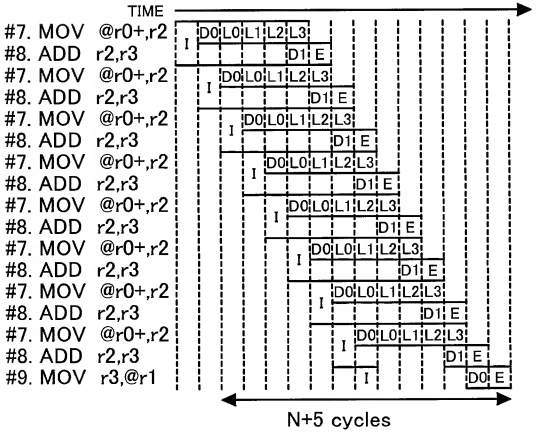

TIME
#7. MOV @r0+,r2
#8. ADD r2,r3
#7. MOV @r0+,r2
#8. ADD r2,r3
#7. MOV @r0+,r2
#8. ADD r2,r3
#7. MOV @r0+,r2
#8. ADD r2,r3
#7. MOV @r0+,r2
#8. ADD r2,r3
#7. MOV @r0+,r2
#8. ADD r2,r3
#7. MOV @r0+,r2
#8. ADD r2,r3
#9. MOV r3,@r1
I
D0 L0 L1 L2 L3
D1 E
N+5 cycles

FIG. 6

```
#1.LDRS _repeat_start   #7.MOV  @r0+,r2   #11. _repeat_start ADD r2,r3      #19.ADD r2,r3
#2.LDRE _repeat_end     #8.MOV  @r0+,r4   #12.               MOV @r0+,r2   #20.ADD r4,r3
#3.LDRC #2              #9.MOV  @r0+,r5   #13.               ADD r4,r3     #21.ADD r5,r3
#4.MOV  #x_addr,r0      #10.MOV @r0+,r6   #14.               MOV @r0+,r4   #22.ADD r6,r3
#5.MOV  #y_addr,r1                        #15.               ADD r5,r3     #23.MOV r3,@r1
#6.MOV  #0,r3                             #16.               MOV @r0+,r5
                                          #17.               ADD r6,r3
                                          #18. _repeat_end   MOV @r0+,r6
```

FIG. 7

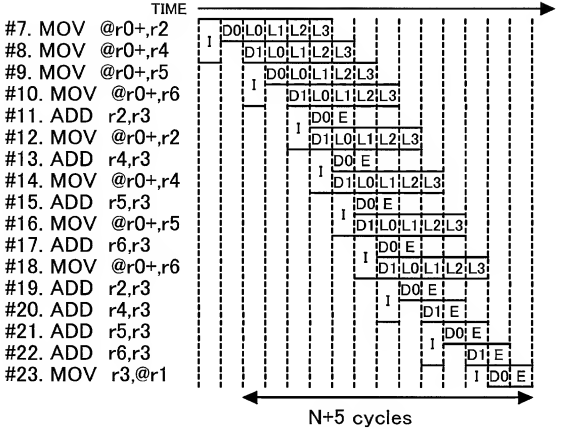


FIG. 8

#1.	MOV #8,r4	#5.MOV	@r0+,r2	#9.	TERM
#2.	MOV #x_addr,r0	#6.ADD	r2,r3	#10. _exit	THABORT
#3.	MOV #y_addr,r1	#7.DT	r4	#11.	MOV r3,@r1
#4. _thread0	FORK _thread0	#8.BT/S	_exit		

FIG. 9

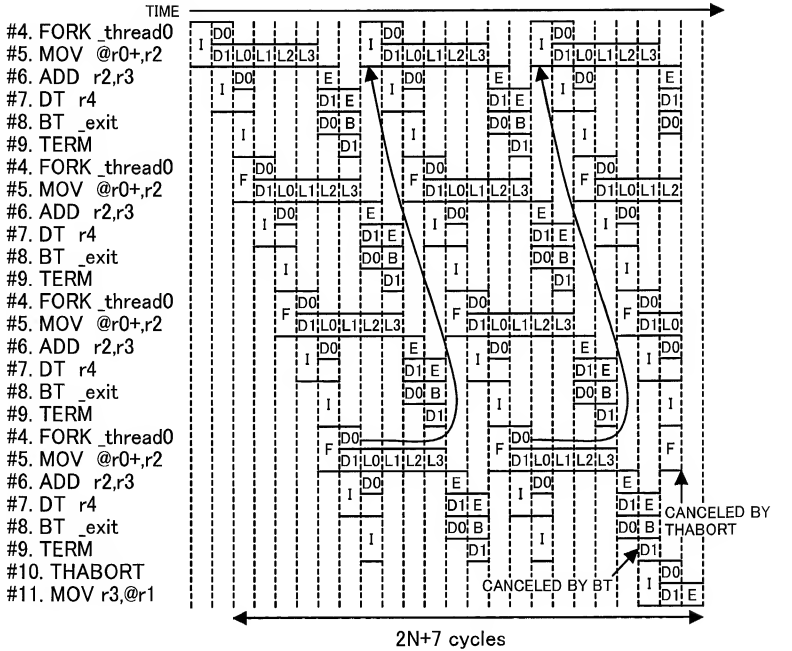

TIME
#4. FORK _thread0
#5. MOV @r0+,r2
#6. ADD r2,r3
#7. DT r4
#8. BT _exit
#9. TERM
#4. FORK _thread0
#5. MOV @r0+,r2
#6. ADD r2,r3
#7. DT r4
#8. BT _exit
#9. TERM
#4. FORK _thread0
#5. MOV @r0+,r2
#6. ADD r2,r3
#7. DT r4
#8. BT _exit
#9. TERM
#4. FORK _thread0
#5. MOV @r0+,r2
#6. ADD r2,r3
#7. DT r4
#8. BT _exit
#9. TERM
#10. THABORT
#11. MOV r3,@r1
CANCELED BY THABORT
CANCELED BY BT
2N+7 cycles

FIG. 10

```
#1.LDRS _repeat_start      #7. _thread0        MOV @r0+,r2
#2.LDRE _repeat_end        #8.                  UNCOND_SUSPEND
#3.MOV #x_addr,r0          #9. _repeat_start   FORK _thread1
#4.FORK  thread0           #10.                 ADD r2,r3
#5.LDRC  #7                #11. _thread1        MOV @r0+,r2
#6.MOV  #y_addr,r1         #12. _repeaat_end UNCOND_SUSPEND
                           #13.                 ADD r2,r3
                           #14.                 MOV r3,@r1
```

FIG. 11

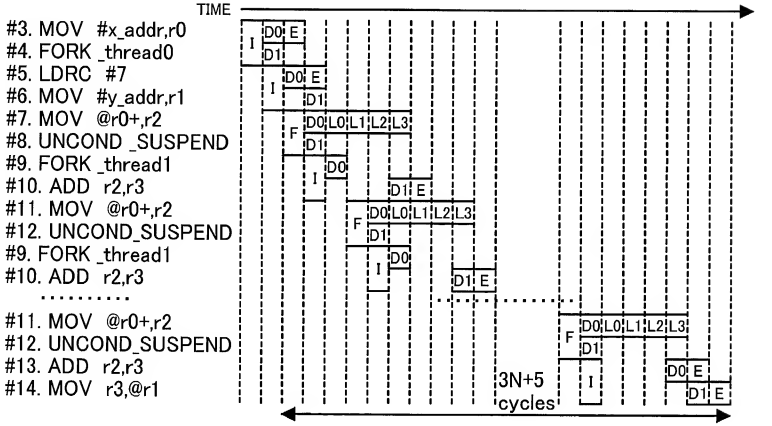

TIME
#3. MOV #x_addr,r0
#4. FORK _thread0
#5. LDRC #7
#6. MOV #y_addr,r1
#7. MOV @r0+,r2
#8. UNCOND _SUSPEND
#9. FORK _thread1
#10. ADD r2,r3
#11. MOV @r0+,r2
#12. UNCOND_SUSPEND
#9. FORK _thread1
#10. ADD r2,r3
.....
#11. MOV @r0+,r2
#12. UNCOND_SUSPEND
#13. ADD r2,r3
#14. MOV r3,@r1
I
D0
E
D1
F
L0
L1
L2
L3
3N+5
cycles

FIG. 12

	DATA NUMBER	LOAD LATENCY	OUT OF ORDER, SORTWARE PIPELINE	MERLOT METHOD	PRIOR ART (JPA8-249183)	CONVENTIONAL PROCESSOR
#1	N	L	N+L+1	MAX(2N+L+2, (L+3)N/4+7)	MAX(3N+L+1, (L-1)N+5)	LN+2
#2		4	N+5	2N+7	3N+5	4N+2
#3		30	N+31	33N/4+7	29N+5	30N+2
#4	8	4	13	23	29	34
#5		30	39	73	237	242
#6	32	4	37	51	101	130
#7		30	63	271	933	962

FIG. 13

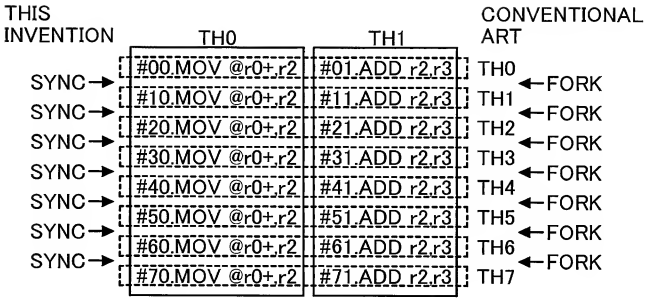

THIS INVENTION
TH0
TH1
CONVENTIONAL ART
SYNC
#00.MOV @r0+,r2
#01.ADD r2,r3
TH0
FORK
#10.MOV @r0+,r2
#11.ADD r2,r3
TH1
#20.MOV @r0+,r2
#21.ADD r2,r3
TH2
#30.MOV @r0+,r2
#31.ADD r2,r3
TH3
#40.MOV @r0+,r2
#41.ADD r2,r3
TH4
#50.MOV @r0+,r2
#51.ADD r2,r3
TH5
#60.MOV @r0+,r2
#61.ADD r2,r3
TH6
#70.MOV @r0+,r2
#71.ADD r2,r3
TH7

FIG. 14

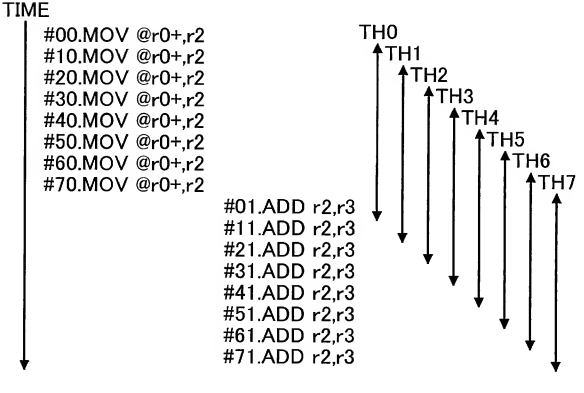

TIME
#00.MOV @r0+,r2
#10.MOV @r0+,r2
#20.MOV @r0+,r2
#30.MOV @r0+,r2
#40.MOV @r0+,r2
#50.MOV @r0+,r2
#60.MOV @r0+,r2
#70.MOV @r0+,r2
#01.ADD r2,r3
#11.ADD r2,r3
#21.ADD r2,r3
#31.ADD r2,r3
#41.ADD r2,r3
#51.ADD r2,r3
#61.ADD r2,r3
#71.ADD r2,r3
TH0
TH1
TH2
TH3
TH4
TH5
TH6
TH7

FIG. 15

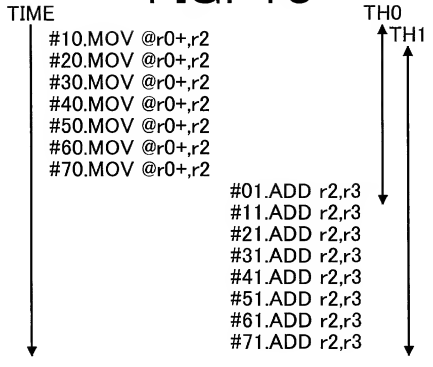


FIG. 16

#1.	LDRE _repeat0	#11._thred1	LDRS _repeat1
#2.	MOV #x_addr,r0	#12.	LDRE _repeat1
#3.	LDRS _repeat0	#13.	LDRC #8
#4.	MOV #y_addr,r1	#14.	NOP
#5.	THRDG/R _thred1	#15._repeat1	ADD r2,r3
#6.	MOV #0,r3	#16.	MOV r3,@r1
#7.	LDRC #8	#17.	THRDE
#8.	NOP		
#9._repeat0	MOV @r0+,r2		
#10.	SYNCE		

THRDG/R: THREAD GENERATION REPEAT TYPE
THRDE : THREAD END
SYNCE : THREAD END SYNCHRONISM

FIG. 17

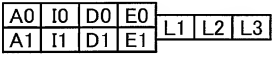


FIG. 18

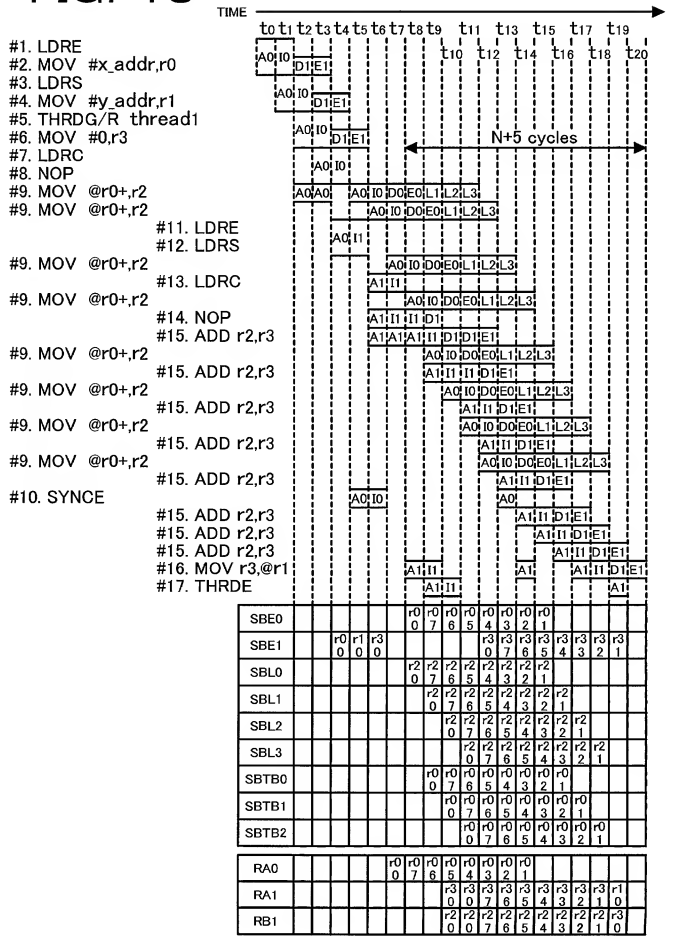

TIME
t0 t1 t2 t3 t4 t5 t6 t7 t8 t9 t10 t11 t12 t13 t14 t15 t16 t17 t18 t19 t20
#1. LDRE
#2. MOV #x_addr,r0
#3. LDRS
#4. MOV #y_addr,r1
#5. THRDG/R thread1
#6. MOV #0,r3
#7. LDRC
#8. NOP
#9. MOV @r0+,r2
#9. MOV @r0+,r2
#11. LDRE
#12. LDRS
#9. MOV @r0+,r2
#13. LDRC
#9. MOV @r0+,r2
#14. NOP
#15. ADD r2,r3
#9. MOV @r0+,r2
#15. ADD r2,r3
#9. MOV @r0+,r2
#15. ADD r2,r3
#9. MOV @r0+,r2
#15. ADD r2,r3
#9. MOV @r0+,r2
#15. ADD r2,r3
#10. SYNCE
#15. ADD r2,r3
#15. ADD r2,r3
#15. ADD r2,r3
#16. MOV r3,@r1
#17. THRDE
N+5 cycles
SBE0
SBE1
SBL0
SBL1
SBL2
SBL3
SBTB0
SBTB1
SBTB2
RA0
RA1
RB1

FIG. 19

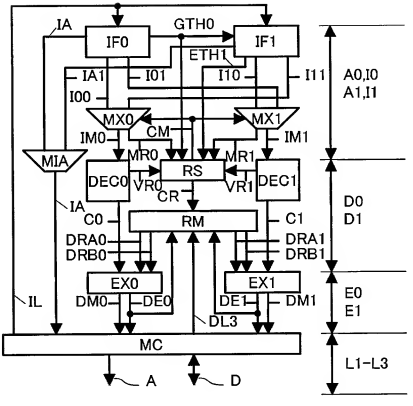

IA
GTH0
IF0
IF1
ETH1
IA1
I01
I10
I11
A0,I0
A1,I1
I00
MX0
MX1
CM
IM0
IM1
MIA
MR0
MR1
RS
DEC0
VR0
VR1
DEC1
CR
D0
D1
IA
C0
RM
C1
DRA0
DRB0
DRA1
DRB1
EX0
EX1
E0
E1
IL
DM0
DE0
DE1
DM1
DL3
MC
L1-L3
A
D

FIG. 20

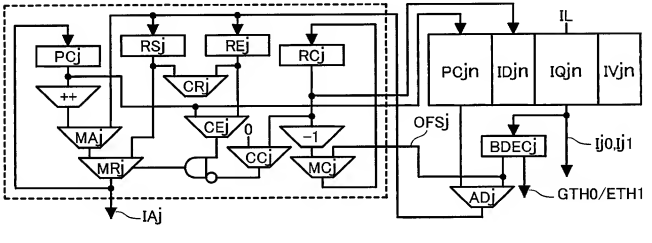

IL
PCj
RSj
REj
RCj
PCjn
IDjn
IQjn
IVjn
++
CRj
MAj
CEj
0
-1
OFSj
BDECj
Ij0,Ij1
MRj
CCj
MCj
ADj
GTH0/ETH1
IAj

FIG. 21

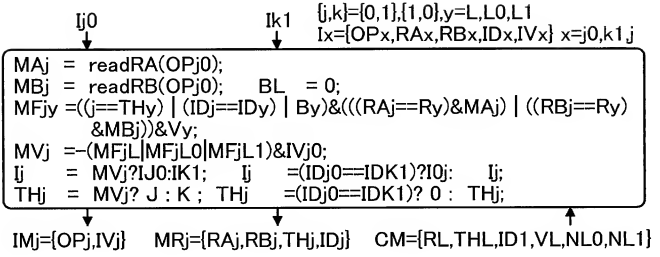
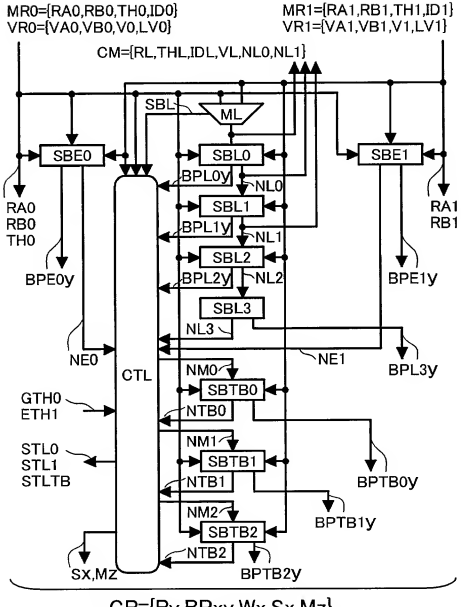


FIG. 22

	EXECUTING ENABLE OR DISABLE		SELECTING INSTRUCTION	
	I00	I10	I0	I1
#1	ENABLE	ENABLE	I00	I10
#2	ENABLE	DISABLE	I00	I01
#3	DISABLE	ENABLE	I11	I10
#4	DISABLE	DISABLE	I11	I01

FIG. 23



CR={Ry,BPxy,Wx,Sx,Mz}

x=E0,E1,L3,TB0,TB1,TB2, y=A0,B0,A1,B1, z=0,1,2

FIG. 24

```
SBL=((TH0==0)&LV0)|((TH0==1)&=-LV1);
if(SBL){RL=RB0;THL=TH0;IDL=ID0;VL=LV0&-STL0}
else    {RL=RB1;THL=TH1;IDL=ID1;VL=LV1&-STL1}
```

FIG. 25

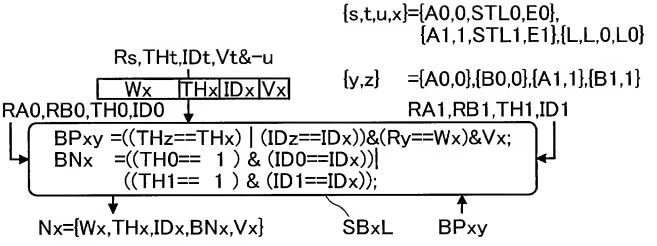

{s,t,u,x}={A0,0,STL0,E0},
{A1,1,STL1,E1},{L,L,0,L0}
Rs,THt,IDt,Vt&~u
Wx THx IDx Vx
{y,z} ={A0,0},{B0,0},{A1,1},{B1,1}
RA0,RB0,TH0,ID0
RA1,RB1,TH1,ID1
BPxy =((THz==THx) | (IDz==IDx))&(Ry==Wx)&Vx;
BNx =((TH0== 1) & (ID0==IDx))|
((TH1== 1) & (ID1==IDx));
Nx={Wx,THx,IDx,BNx,Vx}
SBxL
BPxy

FIG. 26

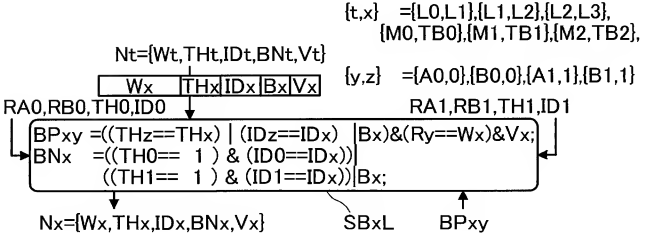

{t,x} ={L0,L1},{L1,L2},{L2,L3},
{M0,TB0},{M1,TB1},{M2,TB2},
Nt={Wt,THt,IDt,BNt,Vt}
Wx THx IDx Bx Vx
{y,z} ={A0,0},{B0,0},{A1,1},{B1,1}
RA0,RB0,TH0,ID0
RA1,RB1,TH1,ID1
BPxy =((THz==THx) | (IDz==IDx) |Bx)&(Ry==Wx)&Vx;
BNx =((TH0== 1) & (ID0==IDx))|
((TH1== 1) & (ID1==IDx))|Bx;
Nx={Wx,THx,IDx,BNx,Vx}
SBxL
BPxy

FIG. 27

```
STL0=((BPL0A0 | BPL1A0 | BPL2A0) & VA0) |
      ((BPL0B0 | BPL1B0 | BPL2B0) & VB0) ;
STL1=((BPL0A1 | BPL1A1 | BPL2A1) & VA1) |
      ((BPL0B1 | BPL1B1 | BPL2B1) & VB1) ;
STL0  |=STL1 & (TH0==1) | ((SBL==1) &LV0) ;
STL1  |=STL0 & (TH0==0) | ((SBL==0) &LV1) ;
STH=  (-GTH0 & STH) | ETH1;
Sx  =  Vx & ((THx==1) |    Bx |    STH)          {x=TB0,TB1,TB2,L3,E0,E1}
Cx  =  Vx & ((THx==0) &- Bx &- STH)
```

STATE						OUTPUTS			
CTB2	CTB1	CTB0	CL3	CE0	CE1	M2	M1	M0	STLTB
*	*	*	0	0	0	TB2	TB1	TB0	0
*	0	*	0	0	1	TB1	TB0	E1	0
0	*	*							
*	0	*	0	1	0	TB1	TB0	E0	0
0	*	*							
*	0	*	1	0	0	TB1	TB0	L3	0
0	*	*							
0	0	*	0	1	1	TB0	E0	E1	0
			1	0	1	TB0	L3	E1	0
			1	1	0	TB0	L3	E0	0
0	0	0	1	1	1	L3	E0	E1	0
OTHERS						TB2	TB1	TB0	1

```
NM2=(M2==TB2)?NTB2:((M2==TB1)?NTB1:((M2==TB0)?NTB0:((M2==L3)?NL3)));
NM1=(M1==TB1)?NTB1:((M1==TB0)?NTB0:((M1==L3 )?NL3 :((M1==E0)?NE0)));
NM0=(M0==TB0)?NTB0:((M0==L3 )?NL3 :((M0==E0 )?NE0 :((M0==E1)?NE1)));
```

FIG. 28

CR={Ry,BPxy,Wx,Sx,Mz,TH0}

(x=E0,E1,L3,TB0,TB1,TB2, y=A0,B0,A1,B1, z=0,1,2)

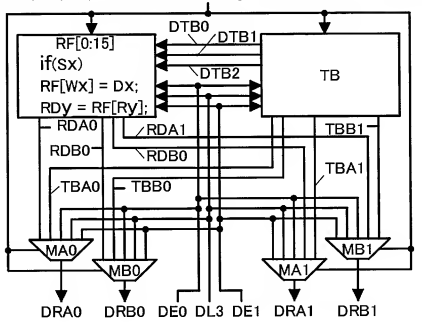


FIG. 29

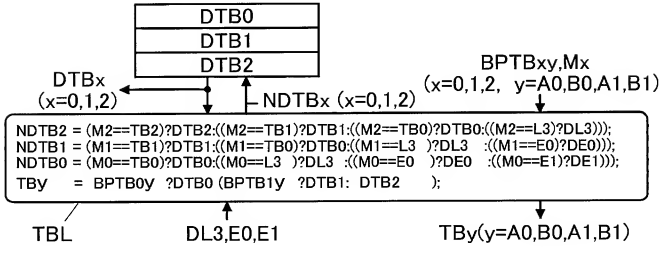


FIG. 30

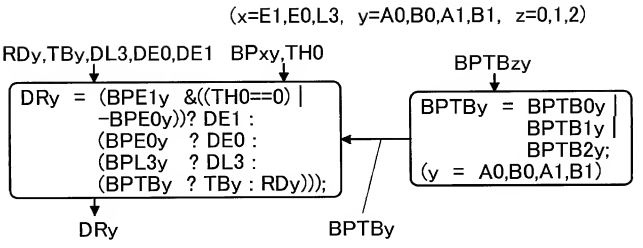
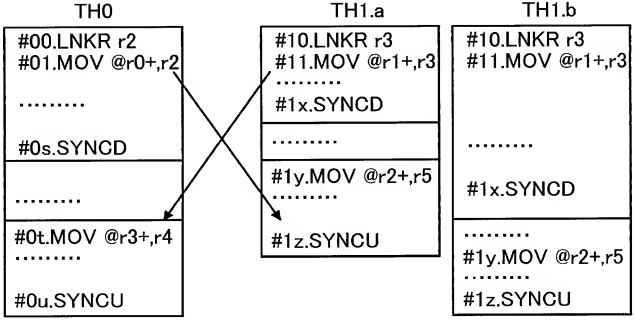

(x=E1,E0,L3, y=A0,B0,A1,B1, z=0,1,2)
RDy,TBy,DL3,DE0,DE1 BPxy,TH0
BPTBzy
DRy = (BPE1y &((TH0==0) |
-BPE0y))? DE1 :
(BPE0y ? DE0 :
(BPL3y ? DL3 :
(BPTBy ? TBy : RDy)));
BPTBy = BPTB0y |
BPTB1y |
BPTB2y;
(y = A0,B0,A1,B1)
DRy
BPTBy

FIG. 31


TH0
#00.LNKR r2
#01.MOV @r0+,r2
.....
#0s.SYNCD
.....
#0t.MOV @r3+,r4
.....
#0u.SYNCU
TH1.a
#10.LNKR r3
#11.MOV @r1+,r3
.....
#1x.SYNCD
.....
#1y.MOV @r2+,r5
.....
#1z.SYNCU
TH1.b
#10.LNKR r3
#11.MOV @r1+,r3
.....
#1x.SYNCD
.....
#1y.MOV @r2+,r5
.....
#1z.SYNCU